Final Exam Preparation - Solutions

Practice Problems

1. Prove that all well-ordered sets are countable.

By definition, every subset of a well-ordered set has a minimum element. We use $\min(S)$ to denote that minimum element. We can define a bijection between the elements of any infinite well-ordered set S and $\mathbb{N}$ as:

$$
\begin{aligned}
0 &\leftrightarrow \min(S_0 := S) \\
1 &\leftrightarrow \min(S_1 := S_0 - \min(S_0)) \\
2 &\leftrightarrow \min(S_2 := S_1 - \min(S_1)) \\
3 &\leftrightarrow \min(S_3 := S_2 - \min(S_2)) \\
&\vdots \leftrightarrow \vdots
\end{aligned}
$$

We are using S_i to denote the remaining subset of S after removing the minimum i elements, so for each natural number we map to the i^{th} least element of S. Because, S is well-ordered this is a unique element. If S is infinite, this means S is countably infinite. If S is finite, $|S|= k$, the mapping end at $k-1$, giving a bijection between S and $\mathbb{N}_k$. In both cases, this shows that S is countable, proving the proposition that all well-ordered sets are countable.

2. The way we defined an execution of a Turing Machine in Class 23 suggested that there could be more than one execution of a Turing Machine, $TM = (S, T \subseteq S \times \Gamma \rightarrow S \times \Gamma \times \textit{dir}, q_0 \in S, q_{Accept} \subseteq S)$, on a given input I. What property of T would ensure that there is only a single execution possible?

A Turing Machine has only a single execution if T is a *function*. For a given input, $(s, \gamma) \in S \times \Gamma$, there is only out possible transition output. We call a Turing machine where T is a function as *deterministic Turing machine*. If T has multiple possible outputs for some input configuration, then it is a *non-determinstic Turing machine*. Recall that our definition of *accepts* is if there is *any* execution of the TM that ends in an accepting state on that input, so this suggest for a non-deterministic TM it is necessary to explore all possible executions to know if an input is accepted. The famous $P = NP$ open problem asks whether there are problems which can be solved efficiently by a nondeterministic Turing machine (*NP* is nondeterministic polynomial time) that cannot be solved efficiently by a deterministic Turing machine (*P* is deterministic polynomial time).

3. Describe a deterministic Turing Machine that never halts but never repeats a configuration.

Just keep moving right - since the tape keeps changing no configuration is repeated, but the Turing Machine runs forever:

$$M = (S = \{0\}, T = \{(0, \text{-}) \rightarrow (0, \mathbf{X}, \mathbf{R})\}, q_0 = 0, q_{Accept} = \{0\})$$

4. Prove that a deterministic Turing Machine that repeats a configuration never halts.

If the TM machine is deterministic, given an input s, γ, there is only output transition. So, if a TM returns to a previous configuration, everything about the machine is in exactly the same configuration, and it does exactly the same thing it did last time. It will repeat whatever sequence of steps it did previously, to return again to that same configuration, and keep doing this forever, without ever halting.

Consider the Turing Machine, M_X, defined below. (The **-** symbol denotes a blank square, which may not appear in the input. Every square to the right of the last input symbol is initially blank.) Hint: for questions 5 and 6 you should be able to use similar techniques to how we reasoned about state machines, but need to also take into account the tape (so instead of using the invariant principle for states as before, now you need to consider it for full machine configurations).

$$\begin{aligned} M_X = (&S = \{A, B, C, D\}, \\ &T = \{(A, \mathbf{0}) \to (B, \mathbf{X}, \mathbf{R}), (A, \text{-}) \to (D, \text{-}, \mathbf{Halt}) \\ &\quad (B, \mathbf{0}) \to (B, \mathbf{0}, \mathbf{R}), (B, \text{-}) \to (C, \text{-}, \mathbf{L}), \\ &\quad (C, \mathbf{0}) \to (C, \mathbf{0}, \mathbf{L}), (C, \mathbf{X}) \to (A, \mathbf{X}, \mathbf{R})\}, \\ &q_0 = A, q_{Accept} = \{D\}) \end{aligned}$$

5. Prove that M_X running on any initial tape with finite input (that is, the number of non-blank squares is $k \in \mathbb{N}$) always terminates.

The machine terminates when it is in state A and the read symbol is **-**. So, we need to show the machine eventually reaches a configuration where the input it $(A, \text{bf -})$, or it gets stuck in some state with no matching output transition.

The high-level operation of the machine is:

– In state A, if the read symbol is **0**, cross it out (write an **X**), enter state B and move right. Otherwise, either the read symbol is **-** and the machine transitions to D and halts, or there is no transition rule and the machine is stuck and halts. So, if we are in state A, the machine either terminates, or enters state B.

– In state B, if the read symbol is **0**, the machine moves to the right, staying in state B and preserving (re-writing) the **0**. This continues, moving to the right past all the **0**s, until the machine reaches a **-**, and transitions to state C. So, when the machine is in state B, if there are any symbols to the right of the current position that are not **0** before the first blank, the machine will get stuck and terminate; otherwise, it ends up in state C, one position to the left of the first **-**.

– In state C, if the read symbol is **0**, the machine moves to the left, staying in state C and preserving the **0**. As with state B, except now moving left, this continues past all the **0** symbols on the tape until an **X** is read. If there is some other symbol, the machine is stuck and terminates. Otherwise, it eventually reaches the **X**, and enters state A. The one important case we should worry about is if the remainder of the symbols on the tape are all **0**s - then, according to our TM transition rules, if it is at the left edge of

the tape and the rule is $(C, \mathbf{0}) \to (C, \mathbf{0}, \mathbf{L})$, it would continue forever in that state staying at the leftmost square. So, for termination to hold, we need to know there is no way the machine could be in state C where everything to the left of the curretnt tape position is a **0**. This follows since $q_0 = A$, and the first rule for A is $(A, \mathbf{0}) \to (B, \mathbf{X}, \mathbf{R})$ — of the leftmost symbol on the tape is a **0**, it is replace with an **X**. Since there is no rule that replaces an **X** with any other symbol, we know that it can never be the case that the machine is in state C and the leftmost tape symbol is an **0**.

So, we know in state A, the machine either terminates or reaches state B; from state B, the machine either terminates or eventually reaches state C, and from state C, the machine either terminates or eventually reaches state A. Each time the machine goes through the $A \to^* B \to^* C \to^* A$ cycle, one of **0** symbols on the tape becomes an **X** (by the $(A, \mathbf{0}) \to (B, \mathbf{X}, \mathbf{R})$ rule), and none of the other symbols on the tape change (for every other rule, the read and write symbols are identical). Since the number of **0** symbols on the tape is initially finite, it must eventually reach zero, and the machine must reach state A. In state A, if the read symbol is not a **0** the machine terminates. Thus, we have proven that for any finite input, the machine always terminates.

6. Prove that $\mathcal{L}(M_X) = \mathbf{0}^*$ (that is, the language recognized by M_X is the set of all strings of zero or more **0** symbols).

From the previous question, we proved the machine always terminates. Now, we need to prove that it terminates in an accept state for input strings of $\mathbf{0}^*$ and in a rejecting state for all other strings.

First, let's prove that no input string that is not of the form $\mathbf{0}^*$ is accepted. If the first input symbol is not a **0**, the machine is stuck in state A and rejects. Otherwise, the machine enters state B and is at the second tape position. In state B, any symbols other than a **0** in the input before the first blank (which must be the end of the input) has no output transition, so the machine is stuck if the input is not all **0**s until the end. Thus, any string that is not $\mathbf{0}^*$ is rejected.

It is left to show that all inputs strings of **0**s are accepted. Informally, we should see that the machine works by crossing off each **0** until it gets to the end, where the read symbol is **-**, where it transitions to state D and halts, accepting the input since $D \in q_{Accept}$.

Let's try to prove it more formally using induction. Our induction predicate is that M_X accepts string of **0**s of length n:

$$P(n) ::= M_X \text{ accepts } \mathbf{0}^n.$$

Base cases: $P(0)$. We need to show M_X accepts $\mathbf{0}^0$ which is the empty string. The machine starts in state A with read symbol **-** (blank, since the input is empty). The rule, $(A, \text{-}) \to (D, \text{-}, \textbf{Halt})$, leave the machine in state D, which is an accepting state.

Induction case: $\forall m.\, \mathbb{N}.\, P(m) \implies P(m+1)$. By the inductive hypothesis, M_X accepts $\mathbf{0}^m$. In order for M_X to accept input $\mathbf{0}^m$, it has an execution that starts from configuration $(A, \star\mathbf{0}^m)$ (we use $\star$ to indicate the position of the tape head; the read symbol is the first to the right of the $\star$) and ends in state D.

We will show that (1) starting from state A and input $\mathbf{0}^{m+1}$ the execution reaches the configuration $(A, \mathbf{X} \star \mathbf{0}^m)$ and (2) will argue that that configuration is equivalent (as far as the machine is concerned) to $(A, \star\mathbf{0}^m)$, so by the inductive hypothesis means the machine accepts.

Part (1): Starting from $(A, \star\mathbf{0}^{m+1}$, the machine takes these transitions: $(A, \star\mathbf{0}^{m+1}) \to (B, \mathbf{X} \star \mathbf{0}^m) \to (B, \mathbf{X0} \star \mathbf{0}^{m-1}) \to^{m-1} (B, \mathbf{X0}^m\star) \to (C, \mathbf{X0}^{m-1} \star \mathbf{0}) \to^{m-1} (C, \mathbf{X} \star \mathbf{0}^m) \to (C, \star\mathbf{X0}^m) \to (A, \mathbf{X} \star \mathbf{0}^m)$

Part (2): As far as M_X is concerned, $(A, \mathbf{X} \star \mathbf{0}^m)$ is equivalent to $(A, \mathbf{0}^m)$. To argue that they are equivalent, we need to know the machine never sees the **X** to the left of the current position. The first step reaches configuration $(B, \mathbf{XX} \star \mathbf{0}^{m-1})$. Now, we have **XX** on the left side of the tape. But, the only rule with read symbol **X** is, $(C, \mathbf{X}) \rightarrow (A, \mathbf{X}, \mathbf{R})$ which moves right (and leaves the **X**). So, any **X** to the left of another **X** is never seen by the machine. Thus, $(A, \mathbf{X} \star \mathbf{0}^m)$ is equivalent to $(A, \mathbf{0}^m)$, so the machine accepts $\mathbf{0}^{m+1}$ by the inductive hypothesis.

(Sorry, my hint about using the invariant principle here was misleading. We ended up using regular induction. There are ways you could use the invariant principle to prove this, and, if we wanted a more formal proof that you reach the end of the **0**s, that might be done using the invariant principle, but I don't think there is a simpler proof using the invariant principle than the one here using regular induction. If anyone came up with a good proof using the invariant principle, please send it to me for "bonus" credit.)

